Automatic and Interactive Mesh to T-Spline Conversion

Wan Chiu Li, Nicolas Ray, Bruno Lévy

Eurographics Symposium on Geometry Processing (2006)
Konrad Polthier, Alla Sheffer (Editors)

Automatic and Interactive Mesh to T-Spline Conversion

Wan-Chiu Li Nicolas Ray Bruno Lévy †

INRIA-ALICE University Nancy 2 INRIA-ALICE

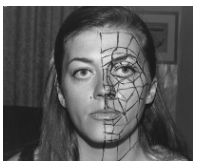
Abstract

In Geometry Processing, and more specifically in surface approximation, one of the most important issues is the automatic generation of a quad-dominant control mesh from an arbitrary shape (e.g. a scanned mesh). One of the first fully automatic solutions was proposed by Eck and Hoppe in 1996. However, in the industry, designers still use manual tools (see e.g. cyslice). The main difference between a control mesh constructed by an automatic method and the one designed by a human user is that in the second case, the control mesh follows the features of the model. More precisely, it is well known from approximation theory that aligning the edges with the principal directions of curvature improves the smoothness of the reconstructed surface, and this is what designers intuitively do.

In this paper, our goal is to automatically construct a control mesh driven by the anisotropy of the shape, mimicking the mesh that a designer would create manually. The control mesh generated by our method can be used by a wide variety of representations (splines, subdivision surfaces . . .).

We demonstrate our method applied to the automatic conversion from a mesh of arbitrary topology into a T-Spline surface. Our method first extracts an initial mesh from a PGP (Periodic Global Parameterization). To facilitate user-interaction, we extend the PGP method to take into account optional user-defined information. This makes it possible to locally tune the orientation and the density of the control mesh. The user can also interactively remove edges or sketch additional ones. Then, from this initial control mesh, our algorithm generates a valid T-Spline control mesh by enforcing some validity constraints. The valid T-Spline control mesh is finally fitted to the original surface, using a classic regularized optimization procedure. To reduce the L^∞ approximation error below a user-defined threshold, we iteratively use the T-Spline adaptive local refinement.

1. Introduction



With the advent of scanning technology, it is now reasonably easy to obtain a computer representation of an existing object. Initiatives such as the `aim@shape` network of excellence [aim] federate research efforts in this direction. However, the pioneer Henri Gouraud often mentioned in his talks that even if it is no longer necessary to draw a mesh onto the "object" to digitize it (see the image of Gouraud's wife, from his Ph.D.), constructing a "good" quad-based control mesh from a shape is one of the most important issues in geometry processing. This was also identified by Malcolm Sabin [Sab] as one of the major challenges in geometry processing. The one million triangle mesh created by a 3D scanner would be completely different from the polygons carefully chosen and drawn by Gouraud on his wife's face. More precisely, due to convergence properties of the approximation [d'A00], it is well known to both skilled designers and researchers in geometry processing [ASD*03] that an efficient quad-based control mesh needs to adapt the anisotropy of the surface. As explained by d'Azevedo, the edges of the control mesh need to be orthogonal (conformality) and aligned with the principal directions of curvature. The existing automatic solutions [EH96] do not meet this requirement, this is why designers still use interactive tools (e.g. [Rap], [Cyb]). Therefore, converting a scanned mesh into an anisotropy-adapted control-mesh remains a user-intensive process.

In this paper, based on recent advances in Geometry Processing [RLL*06], we present a new method to automatically

† {wan-chiu.lilray|levy}@loria.fr

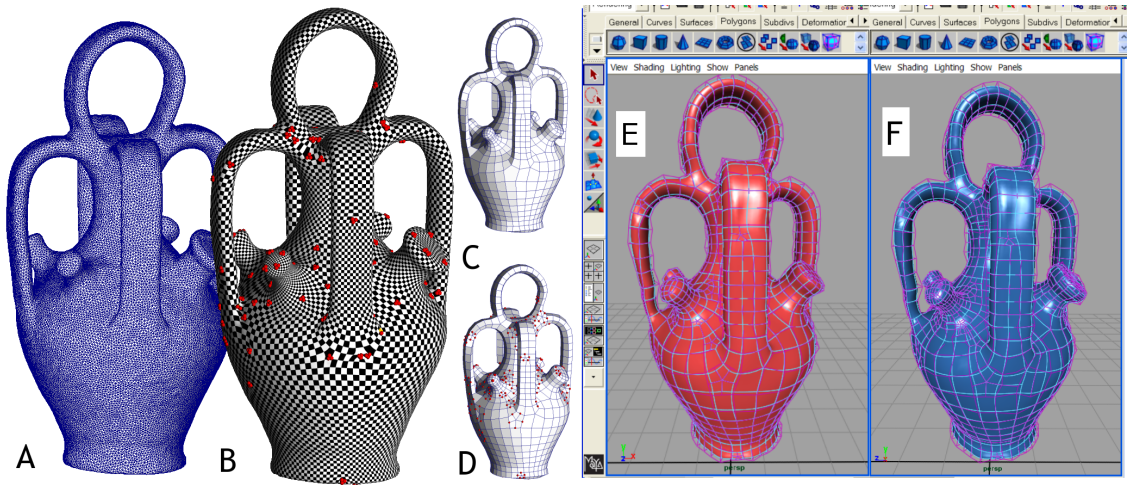


Figure 1: *A: initial triangulated surface; B: PGP (periodic global parameterization). singularities are indicated in red; C: initial control mesh, extracted from the PGP; D: valid T-Mesh; E: T-Spline fitted to the original surface; F: L^∞ fitting with adaptive local refinement.*

construct a quad-dominant control-mesh from a triangulated mesh. In addition, the automatically-constructed control-mesh can be interactively modified by the user. For instance, the user may want to add some control points for further free-form editing. Our method may be used for constructing a wide variety of representations (B-Splines, Nurbs, Catmull-Clark). In the frame of this paper, we demonstrate the automatic conversion of scanned meshes into the T-Spline representation [SZBN03].

Our method is composed of the following steps:

- **extract a quad-dominant control mesh:** We use state-of-the-art methods, such as PGP (Periodic Global Parameterization) shown in Figure 1-B, and extract the control mesh from it (Figure 1-C);
- **manually update the control mesh (optional):** we extend PGP to take into account user-defined information. We also show how the control-mesh may be manually edited, by interactively editing geodesics traced on the original surface;
- **enforce control-mesh validity constraints for T-Splines:** the control mesh of a valid T-Spline is supposed to satisfy some validity conditions. The control mesh obtained at the previous step is transformed to satisfy these conditions (Figure 1-D);
- **fit the control-mesh to the surface:** this is achieved by a classical regularized fitting procedure. The result is fully compatible with readily available industrial software (see Figure 1-E). To minimize the L^∞ norm below a user-defined threshold, we perform adaptive local refinement (see Figure 1-F).

The paper is organized as follows. The remainder of this introduction reviews the previous work. Section 2 shows how to extract an initial control mesh, and presents some tools to interactively modify it. Section 3 explains how to convert this initial control mesh into a valid T-Spline/T-NURCC control mesh (or T-Mesh). The fitting procedure is then explained in Section 4. The paper concludes with some results and suggestions for future work.

Previous work

This section reviews the previous work related with the different steps of our algorithm, control-mesh construction, parameterization and fitting.

1. Quadrilateral control mesh

Manual methods: The trivial solution to obtain a quad-remeshed version of the surface is to draw the boundary curves on the mesh manually as proposed in [KL96] and [MBVW95]. [LLP05] proposed a method with combinatorial data structure which facilitates this kind of curve drawing tasks on meshes. Some commercial softwares, for example, Rapidform [Rap] and Cyslice by Cyberware [Cyb] provide skill designers with tools to patche the objects manually. To provide even more efficient processing, some of them even provide templated-patching for objects with similar shapes and genuses. However, manual patching still requires skilled 3D model designers to obtain a satisfactory effect.

Regular and semi-regular methods: By using a cut-graph that turns a surface into a topological disk, which is then parameterized into a square domain, one can obtain a fully regular control mesh by resampling the geometry im-

age [GGH02]. To produce semi-regular quadrilateral control meshes, [EH96] employed the technique of triangle merging. [BMRJ04] proposed a method based on discrete Lloyd relaxation. Most recently, [DBG*06] have proposed an algorithm that is based on the fact that the Morse-Smale complex induced by any piecewise linear function quadrangulates the surface.

Quad-dominant methods: Some methods consider the anisotropy of the surface since it is optimum from a function approximation point of view. [ASD*03] proposed quad-dominant remeshing method which adapts the anisotropy of the object by explicit integration of the curvature tensor through a parameterization of the surface. Later, [MK04a] improved the method by doing the integration directly on the surface without the need of the parameterization. Nonetheless, explicit integration of stream lines is always plagued by the problem of the seeding and placement of the stream lines. [DKG05] used mixed implicit/explicit schemes with harmonic functions to obtain the control mesh. Finally, [RLL*06] proposed a method that produces the quad-dominant control mesh through the calculation of two periodic functions by implicit integration of a pair of orthogonal vector fields (for example, the two eigenvector fields of the curvature tensor). The quad-dominant control mesh and the parameterization emerge simultaneously from the global numerical optimization process. Seeing the advantages of this method, we have used it to generate our initial quad-dominant control mesh. A more detailed review of this method will be given in Section 2.

2. Global parameterization Fitting a parametric representation to a mesh is much easier if the mesh is parameterized. The first class of methods considers a texture atlas, with multiple charts. Special care needs to be taken considering the smoothness of the parameterization when crossing a chart boundary. In [KLS03], all the charts are simultaneously optimized, with respect to an energy functional taking into account inter-chart transition functions. A similar approach is used in [THCM04]. The second class of methods considers a single "global" parameterization. Gu *et. al*'s approach [GY03] computes the *conformal structure* of a surface. Gortler *et. al* proposed in [GGT04] a general *discrete one-forms* formalism to study this type of methods and prove their validity. Ray *et. al* propose in [RLL*06] the PGP method (Periodic Global Parameterization). We used this latter one, as explained in Section 2.

3. Fitting B-Splines are the most popular representation in CAD/CAM. However, local refinement cannot be done with this representation, since a single control point cannot be inserted without propagating an entire row or column of control point, as done in [EH96]. Surface approximation with subdivision surfaces was also studied [LMH00, LLS01]. These methods minimize some L^∞ errors by subdividing the control mesh *globally*. In order to perform adaptive local insertion of control vertices so as to achieve minimization

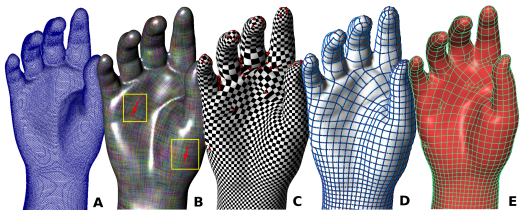


Figure 2: *A: Original surface; B: smoothed control vector field $\vec{K}$ and $\vec{K}^\perp$, interpolating two user-defined directional constraints; C: periodic global parameterization. Singularities are shown in red; D: initial control-mesh; E: A valid T-Mesh.*

of some L^∞ errors, [MK04b] proposed to use Loop subdivision surface with a *local* adaptive refinement procedure. Although this method give satisfactory results, the problem of surface approximation with adaptive local refinement using spline surface is remain unsolved. In the specific case of terrains, T-spline surface fitting to Z-Map models was studied in [ZWS05]. In this paper, we study the T-Spline fitting problem with surfaces of arbitrary topology. Note that other CAD/CAM representations with local refinement capability exist [FB88] (Please refer to [SZBN03] for a summary of the similar methods). Among the possible representations, we chose T-Splines since they can be easily converted to NURBS and subdivision surfaces, and since they offer interesting local refinement capabilities (see further).

2. Creating a Control-Mesh

2.1. Review of Periodic Global Parameterization

To automatically compute an initial quad-dominant control mesh, we use the PGP method described in [RLL*06]. This section gives a quick overview of the method. Given a surface S, two orthogonal unit vector fields $\vec{K}$, $\vec{K}^\perp$ (in practice, $\vec{K}^\perp$ can be determined from $\vec{K}$ and the surface normal), and a user-defined preferred quad edge length ω, PGP generates two periodic functions θ and ϕ defined over S, with their gradients aligned to $\vec{K}$ and $\vec{K}^\perp$ respectively. The control mesh will be then found by extracting certain level sets of θ and ϕ, as explained in the next section.

In our case, since we want to generate an anisotropy-adapted control mesh, the control vector fields $\vec{K}$ and $\vec{K}$ are obtained by smoothing an estimate of the principal directions of curvatures, as in [ASD*03].

The functions θ and ϕ are found by minimizing the following energy functional:

$$F_{PGP} = \int_S \left(\|\nabla\theta - \omega\vec{K}\|^2 + \|\nabla\phi - \omega\vec{K}^\perp\|^2 \right) dS \quad (1)$$

This method is similar to the quad-dominant remeshing in [DKG05], with the two following differences. First, we take

into account the geometry of the surface, through the control vector fields $\vec{K}$ and $\vec{K}^{\perp}$. Then, by optimizing F_{PGP} with respect to alternative variables $(u,v) = \cos(\theta), \sin(\theta)$ (resp $(u',v') = \cos(\phi), \sin(\phi)$), we can use implicit integration for both directions, which avoids the uneven spacing of edges and open loops encountered with explicit integration. The resulting energy is a sum of quadratic terms, defined on the edges of the triangulation:

$$\begin{aligned} F_{PGP} &= \textstyle\sum_{i,j\in E} F_{edge}(i,j) \\ F_{edge}(i,j) &= \left\| \begin{pmatrix} u_i \\ v_i \end{pmatrix} - \begin{pmatrix} \cos(\beta_{ij}) & -\sin(\beta_{ij}) \\ \sin(\beta_{ij}) & \cos(\beta_{ij}) \end{pmatrix} \begin{pmatrix} u_j \\ v_j \end{pmatrix} \right\| \\ \vec{K}_{ij} = 0.5(\vec{K}_i + \vec{K}_j) \quad &; \quad \beta_{ij} = \omega \vec{K}_{ij} \cdot \vec{e}_{ij} \end{aligned} \tag{2}$$

where, $E = \{\vec{e}_{ij}\}$ is the set of edges of the mesh from the ith to the jth vertex. To minimize F_{PGP}, we fix one of the vertices $u_1 = 1, v_1 = 0, u'_1 = 1, v'_1 = 0$ and minimize F_{PGP} with respect to all the other variables. Since F_{PGP} is a quadratic form, this means solving a sparse symmetric system. We use the conjugate gradient algorithm with Jacobi's preconditioner. The reader is referred to [RLL*06] where some refinements of the method are explained, such as coupling θ and ϕ, modulating the edge length ω, or using an improved solution mechanism for large meshes.

Algorithm 1 Propagation from (θ_i) to (θ_j) along the edge (i,j)

$$\begin{aligned} &\theta_j \leftarrow \arctan(v_i/u_i) \quad ; \quad \vec{n} \leftarrow \vec{e}_{ij}/\|\vec{e}_{ij}\| \\ &s \leftarrow \text{argmin}_s \left| \theta_i - (\pi/\omega)\vec{n} \cdot (\vec{K}_i + \vec{K}_j) - \theta_j + 2s\pi \right| \\ &\theta_j \leftarrow \theta_j + 2s\pi \end{aligned}$$

Once the alternative variables $(u,v) = (\cos(\theta), \sin(\theta))$ are computed (resp. $(u',v') = (\cos(\phi), \sin(\phi))$), once needs to compute the corresponding variables θ (resp. ϕ). To achieve this, we need to disambiguate the value of θ among all the possible values $\theta + 2k\pi$.

This is done in each triangle individually, by starting at a given vertex of the triangle $\mathbf{p_1}$ with $\theta = \arctan(v_1/u_1)$, and propagating along the three edges, using Algorithm 1. This algorithm chooses among all the possible values of θ_j the one nearest to the optimum value $\theta_i + \beta_{ij}$, where $\beta_{i,j} = \omega \vec{K}_{ij} \cdot \vec{e}_{ij}$ corresponds to the optimum displacement along the edge (i,j).

Once we have reconstructed the parameterization in each individual triangle, we need to check some validity conditions. More precisely, angles around a vertex should sum to 2π, angles around a triangle should sum to π, and vertex neighborhoods should satisfy the wheel compatibility condition (see e.g., [SdS01]). Vertices and triangles that violate these conditions will be referred to as *singular* in what follows.

2.2. Extracting the Control Mesh

Once we have computed the local parameterization in each triangle T, we construct the chart layout. In our setting, the chart boundaries are defined to be the iso-$2k\pi$ lines of θ and ϕ. This defines a set of segments in each triangle. Note that if a triangle T is traversed by an iso-$2k\pi$ line of θ (resp. ϕ), this triangle translated by $2s\pi$ will be traversed at the same location by the iso-$2(k+s)\pi$ line of θ (resp. ϕ). As a consequence, the extremities of these independent segments match along the edges of the triangulation, and the segments form continuous polygonal lines.

Algorithm 2 Control-mesh layout

```
compute chart boundaries:
for each triangle T
  if T is non-singular
    for k ∈ ℕ such that 2kπ ∈ [min_T(θ), Max_T(θ)]
      Line l ← line of equation(θ = 2kπ)
      Segment S ← l ∩ T //in parameter space
      store S in T
      store the extremities of S in the corresponding edges of T
    end//for
    repeat the above procedure for φ
  end//if
end//for
for each edge e
  merge the segment extremities stored in e
  that share the same geometric location in 3D
end//for
for each triangle T
  compute the intersections between the edges stored in T
end//for
remove all dangling segments
```

Algorithm 2 gives the general algorithm to extract the initial control-mesh layout. Dangling segments, namely, segments with a free extremity, are caused by the singular vertices and triangles.

A key aspect in this algorithm is to have an efficient data structure to represent a set of lines embedded in a surface. We used the embedded simplicial complexes presented in [LLP05] with provably optimum complexity for all the operations. For instance, by storing in each triangle T the set of segments contained by T, it computes all the intersections between the $iso-\theta$ and $iso-\phi$ curves with linear complexity in the total number of triangles. This data structure will also be used in the next section, to facilitate the interactive editing of the control mesh.

2.3. Interactive Editing (*Optional*)

The so-constructed control mesh can be interactively edited by the user. First, it is possible to add constraints in the

vector-field smoothing algorithm used to construct the control vector fields $\vec{K}$ and $\vec{K}^{\perp}$. Second, once the initial control-mesh is represented by an embedded simplicial complex [LLP05], all the related manual editing operations and design with geodesic curves can be applied.

Constrained vector field smoothing: To allow the user to interactively edit the control vector field, we use a variant of Hertzmann *et. al*'s method [HZ00]. In this method, the vector field is represented by a set of angles α relative to an initial arbitrary tangent vector field $\vec{t}$. We use a weighted sum of a fitting term and smoothing term.

$$R = (1-\rho)\underbrace{\sum_i \left\|\alpha_i - \alpha_i^0\right\|^2}_{\text{fitting term}} + \rho\underbrace{\sum_{\vec{e}_{ij}} \left\|\alpha_i - \alpha_j - \text{angle}(\vec{t}_i,\vec{t}_j))\right\|^2}_{\text{smoothing term}} \tag{3}$$

where the angles α_i^0 are computed from the initial values of the vector field $\vec{K}$ at the vertices and angle$(\vec{t}_i,\vec{t}_j)$ is defined in the range $]-\pi,\pi]$. The user-defined coefficient ρ corresponds to the desired smoothing intensity (in all our examples, $\rho = 0.8$). The smoothing term minimizes the variations of α over an edge $\vec{e}_{ij}$. As before, to make the variable independent of $2k\pi$ translations, we solve for the sines and cosines. User-prescribed directions are simply removed from the degrees of freedom. Figure 2-B shows how two user-defined directional constraints are interpolated.

Manual editing and geodesic design: Using the embedded simplicial complex data structure, all the possible relations between the initial triangulated surface and the control mesh under construction are represented with optimum access time. This makes it easy to implement the following operations (see Figure 3 and the companion video):

- edge straightening: an edge of the control mesh is replaced with a geodesic;
- edge insertion: a new edge is created between two points picked on the surface. All the intersections are kept up-to-date;
- edge deletion.

3. Constructing a T-Spline/T-NURCC

After the automatic and optional interactive steps described in the previous section, we obtain a quad-dominant control mesh. However, our target T-Splines representation has some requirements characterizing valid control meshes (or T-Meshes) that our control mesh might violate. This section presents an automatic algorithm to enforce these constraints in our control mesh. In addition, we show how to compute the knot-intervals that define the function basis attached to the T-Mesh.

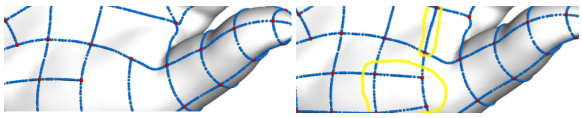


Figure 3: *Manually editing the initial control mesh by adding geodesics (circled). Note that all the intersections are kept up-to-date.*

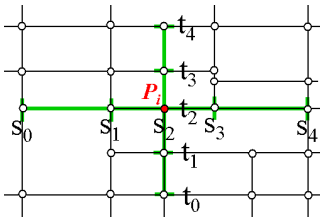


Figure 4: *The pre-image of a T-Mesh*

3.1. T-Splines

The main limitation of CAD/CAM representations is their lack of flexibility, and the highly constrained nature of their admissible control meshes. For instance, the control-mesh of a standard B-Spline surface needs to be locally equivalent to a regular grid (a cylinder and a torus are the only possible variations). To overcome this limitation, T-Splines were proposed by Sederberg *et. al* in [SZBN03].

The control points of a T-spline surface are arranged by means of a control grid called a *T-Mesh*, where local refinements can be done. If a T-Mesh forms a rectangular grid, the T-Spline degenerates to a B-spline surface. Figure 4 shows the pre-image of a portion of a T-Mesh. However, despite their support for local refinement and T-junctions, T-Splines cannot have vertices of arbitrary valence in their control mesh. For this reason, Sederberg *et. al* also proposed in [SZBN03] the T-NURCC (Non-Uniform Rational Catmull Clark) representation, that can fill-in the neighborhoods of the extraordinary vertices, while smoothly connecting with the rest of the T-Splines.

In a T-Spline, each edge of the T-Mesh has an associated *knot interval*, that needs to satisfy the following two rules:

Rule 1: The sum of the knot intervals on opposing edges of any face must be equal.
Rule 2: If two T-junctions on opposing edges of a face can be connected without violating the previous rule, that edge must be included in the T-Mesh.

Given a valid T-Mesh, the equation of a T-Spline is defined by:

$$\mathbf{S}(s,t) = \frac{\sum_{i=1}^{n} w_i P_i B_i(s,t)}{\sum_{i=1}^{n} w_i B_i(s,t)}, \quad s,t \in D \tag{4}$$

where: P_i is the ith control point (x_i, y_i, z_i), $i = 1 \cdots n$. The blending function $B_i(s,t) = N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](s) N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](t)$, where, $N[s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}](s)$ is the cubic B-spline basis function associated with the knot vector $s_i = [s_{i0}, s_{i1}, s_{i2}, s_{i3}, s_{i4}]$ and $N[t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}](t)$ is associated with the knot vector $t_i = [t_{i0}, t_{i1}, t_{i2}, t_{i3}, t_{i4}]$.

The two knot vectors s_i and t_i of the two basis functions of the control point P_i are inferred from the knot information of the T-Mesh as follows (see Figure 4): Let (s_{i2}, t_{i2}) are the knot coordinates of P_i. Consider a ray in parameter space $R(\alpha) = (s_{i2} + \alpha, t_{i2})$. Then s_{i3} and s_{i4} are the s coordinates of the first two s-edges intersected by the ray in the positive α direction. A s-edge is a vertical line segment of constant s. The s_{i0}, s_{i1} and t_i are found similarly.

We first ensure T-Spline validity almost everywhere, and "push" the problems by generating new extraordinary vertices when this cannot be avoided (the neighborhood of these extraordinary vertices will be replaced by T-NURCCes, as explained in the next section):

1. whenever possible, enforce Rule 2 by inserting the missing edges in each loop;
2. for all remaining invalid loops (including N-sided loops), convert them into extraordinary vertices as explained below (see also Figure 5).

- A: In each even N-sided loop, a new vertex is inserted and connected to every two vertex of the loop. This creates N/2 additional quads and one extraordinary vertex;
- B: In each odd N-sided loop, a new vertex is inserted and connected to a new T-vertex in each edge of the loop. This creates N additional quads, N T-vertices and one extraordinary vertex;
- C: similarly, odd N-sided loops with a vertex of valence 3 on the border are remeshed as shown in Figure 5.

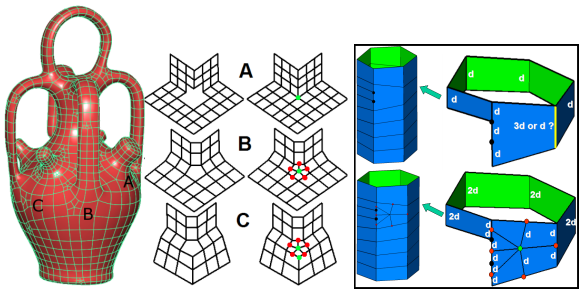


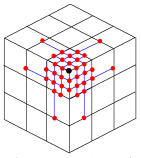
Figure 5: *Left: from N-sided polygon to extraordinary vertices; Right: consistent knot interval assigning through the use of extraordinary vertices.*

Each time an edge is subdivided, its associated knot interval is divided by two. These three operations guarantee that knot intervals remain coherent (see Figure 5-Right).

After this step, the control mesh is a valid T-Spline control mesh everywhere, except in the neighborhood of extraordinary vertices. The concerned quads are replaced with T-NURCCes, as explained in the next section.

3.2. Extraordinary vertices and T-NURCCes

A T-NURCC is a NURCC, which is a modification of cubic NURSSes [SZSS98], with T-junction in the spirit of T-splines. NURCCes is a generalization of both tensor product non-uniform B-spline surfaces and Catmull-Clark surfaces. It enforces the constraint that opposing edges of each four-sided face have the same knot interval. The enforcement of this constraint makes the local subdivision of NURCCes possible. In our implementation, we conceptually apply to each extraordinary vertex two steps of local subdivision (the figure shows a valence-3 vertex). This generates the additional control points marked in red, and expressed by linear combinations of the initial control points. The coefficients of these linear combinations (that depend on the valence of the vertex) are given in Sederberg *et. al*'s paper (and not repeated here for paper length considerations). In practice, we keep a version of the original mesh, and apply local subdivision to a copy. While applying the subdivisions, we store in the newly created control points the list of original control point they depend on together with the coefficients. This representation can be directly used in the subsequent fitting steps, as explained in the next section.



4. Fitting

For surface approximation, in order to measure a defined error metric, one needs to have a correspondence between the approximating and the original surface. Parameterization-free methods [MK04b, DIS03], mostly meant to fit point clouds, geometrically project each sample onto the approximating surface. Parameterization-based methods establish the correspondence by parameterizing the original surface and then identifying the parameter values. Our approach belongs to this latter class of methods.

Given a parameterization $(s,t) \mapsto \mathbf{S}(s,t) \in \mathbb{R}^3$ of the surface, we will minimize the following energy functional, as done in classical regularized fitting methods (see e.g. [Gre94]):

$$E = E_{fit} + \sigma E_{fair}$$

$$\text{where:} \begin{cases} E_{fit} &= \int \|\mathbf{S}(s,t) - \mathbf{M}(s,t)\|^2 ds dt \\ E_{fair} &= \int \left((\frac{\partial^2 \mathbf{S}}{\partial s^2})^2 + 2(\frac{\partial^2 \mathbf{S}}{\partial s \partial t})^2 + (\frac{\partial^2 \mathbf{S}}{\partial t^2})^2 \right) ds dt \end{cases} \quad (5)$$

In this equation, $\mathbf{M}(s,t)$ denotes a parameterization of the

original triangulated surface. As often done in Splines fitting, we approximate the fitting term E_{fit} by using a discrete set of m samples:

$$E_{fit} \simeq \sum_{k=1}^{m} \|\mathbf{S}(s_k,t_k) - (x_k,y_k,z_k)\|^2$$

For each sample (x_k,y_k,z_k) of the original surface, (s_k,t_k) denotes its coordinate in parameter space. The natural idea would be to use the original vertices of the surface, but it is better to re-sample it so that the operation is less sensitive to the resolution of the mesh. The re-sampling is done by using a regular grid of samples in each face in the parameter space (we used 10×10 samples per face in our implementation). The corresponding points on the surface is found easily by linear interpolation in the facets.

The thin-plate energy E_{fair} avoids wiggles of the spline surface. However, if the coefficient σ is set to be too large, the final spline surface may fit less to the original surface. In our examples, we used $\sigma = 0.05$.

Note that by construction, our control mesh and associated knot vector defines a standard (or semi-standard) T-Spline. Therefore, the denominators of the T-Spline is identically one, and we can focus on the numerator:

$$\mathbf{S}(s,t) = \sum_{i=1}^{n} \mathbf{P}_i \mathbf{B}_i(s,t)$$

Each coordinate x,y,z can be processed independently. For the x coordinate, the fitting term is given by:

$$E^x_{fit} = \sum_{k=1}^{m} \left(\sum_{i=1}^{n} X_i \mathbf{B}_i(s_k,t_k) - x_k \right)^2 \qquad (6)$$

where X_i (resp Y_i, Z_i) denote the coordinates at the control point $\mathbf{P}_i$. The X_i's that minimize Equation 6 are also the solution of a linear system $A^tAX = A^tb$, where the coefficients of the $m \times n$ matrix A are given by $a_{k,i} = B_i(s_k,t_k)$ and right-hand side by $b_k = x_k$. The unknown vector X corresponds to all the x coordinates of the control points. Adding the fairing term E_{fair}, the linear system becomes:

$$\left(A^tA + \sigma(A^t_{ss}A_{ss} + 2A^t_{st}A_{st} + A^t_{tt}A_{tt})\right) X = A^tb \qquad (7)$$

where the coefficients $(.{}_{k,i})$ of the $m \times n$ matrices A_{ss},A_{st},A_{tt} are the second order derivatives $B_{iss}(s_k,t_k), B_{ist}(s_k,t_k)$ and $B_{itt}(s_k,t_k)$ of the basis functions B_i respectively.

To solve the regularized fitting problem, the remaining two difficulties are to determine the parameters (s_k,t_k) associated with the vertices of the original surface, and then accumulating the contributions of all the basis functions to construct the matrices A,A_{ss},A_{st},A_{tt} and the right-hand-side b. These two issues will be explained in the next two sections.

4.1. Computing the parameters (s_k,t_k)

The parameterization of the original mesh is obtained as follows. As explained in the previous section, the control mesh is composed of a set of quadrilaterals. By keeping the relation between the control mesh and the surface, it is possible to retrieve in each loop of the control mesh the set of triangles it contains (using a greedy algorithm). Then, two different cases occur, according to the presence of singularities (see Section 2).

- *none of the triangles and vertices is singular:* a parameterization can be easily reconstructed from the (θ,ϕ) values, by greedily assembling the triangle in parameter-space (see [RLL*06], [SdS01]);
- *the loop contains singular triangles and/or vertices:* we re-parameterize the interior of the loop using [Flo03].

Note that all the loops with edges modified by the user are considered to belong to the second category. After all the loop interiors are parameterized, we need to improve cross-boundary continuity. The charts that did not contain singularities already have a globally smooth parameterization. Therefore, we simply apply a relaxation procedure to the re-parameterized charts and their neighbors, by optimizing the smoothness of transition functions, as explained in [SPPH04].

4.2. Constructing and solving the linear system

To solve our regularized fitting problem (Equation 5), the most natural way would be to proceed on a patch-by-patch basis. This would traverse the matrices A,A_{ss},A_{st},A_{tt} row by row, and would make it possible to directly construct the final matrix of the linear system without storing these intermediate matrices.

However, constructing the pre-images of each patch is non-trivial. For this reason, we prefer to iterate on the control nodes. This means we consider one basis functions $\mathbf{B}_i(s,t)$ at a time, with a simpler pre-image. As a consequence, we store the matrices A,A_{ss},A_{st},A_{tt} and construct them column by column. After the traversal of all basis functions, the final matrix of the system is finally assembled (see Equation 7).

The basis functions are piecewise defined in a neighborhood around the pre-image of each control point $\mathbf{P}_i$ (see Figure 4). Each basis function $\mathbf{B}_i$ is completely defined by the T-Mesh and associated knot vectors around the control point $\mathbf{P}_i$. To retrieve them, we first fix arbitrary coordinates (s_0,t_0) to P_i and greedily propagate the knot intervals around it until the region of influence $D_i = [s_{i0},s_{i4}] \times [t_{i0},t_{i4}]$ is completely determined. The pre-image looks like the one shown in Figure 4.

According to this local parameterization, P_i influences the T-spline patches that correspond to the faces intersected by D_i. The patch of each face is mapped to $[0,1]^2$ with $(0,0)$ set at a corner of the face. Therefore, when the influence of P_i is added to the matrices, its pre-image needs to be pre-scaled accordingly. For example, in the pre-image of the T-Mesh, if the size of the rectangle of an influenced face, F, are d and e

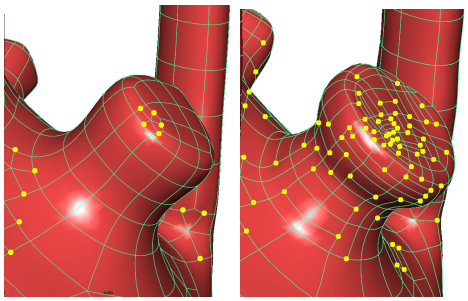


Figure 6: *Adaptive local refinement splits some faces to better capture complex geometry.*

in the s and t directions respectively, the s and t knot vectors of P_i with respect to F should be scaled by $1/d$ and $1/e$ respectively. Once the knot-vectors and region of influence D_i of the basis function B_i are determined, we update the corresponding column in the matrices A, A_{ss}, A_{st} and A_{tt}. After all control points are processed, the final matrix and right-hand side of the system are constructed. All the matrices are represented by column-major sparse data structures (CCS format, for Column Compressed Storage). We use the readily sparse direct solver TAUCS. As a consequence, the inverse of the matrix can be reused to find the X, Y and Z components of the control mesh coordinates. Note that sparse direct solvers perform so well that inversing the matrix is faster than solving a linear system with preconditioned conjugate gradient (see [BBK05], [Lev05] and the timings in the results section).

4.3. Adaptive L^∞ fitting

Global L^2 fitting operates with a fixed number of control points and thus a fixed degree of freedom is sometimes not sufficient to reconstruct the original surface. Therefore, more degree of freedom must be added. Generally, this is done by global refinement of the control mesh, which adds superfluous control points to already low approximation error regions. On the contrary, since we are using T-splines with support for local refinement, new control points can be inserted *locally* in regions of high approximation error (see Figure 6). Thanks to the local support of T-splines, there is no need to carry out the global L^2 fitting every time a new control point is added. Only a smaller linear system needs to be solved involving only the patches of the control mesh affected by the local refinement operation. The L^∞ metric is defined as follows:

$$L^\infty(\mathbf{S},\mathbf{M}) = max_S \|(\mathbf{S}(s,t) - \mathbf{M}(s,t))\|^2 \qquad (8)$$

where $\mathbf{M}(s,t)$ denotes a parameterization of the original surface. This error metric is evaluated by regularly sampling the parameter space of each face.

We iteratively apply the local refinement procedure described below to the face of worst L^∞ approximation error until it drops below a user-defined threshold.

The local refinement of T-spline is one of its invaluable properties. It is also called local knot insertion (please see [SCF*04] for more details). New control points are inserted into the T-Mesh without changing the geometry of the original T-spline surface. The algorithm recovers the T-Spline validity constraints (Section 3) by iteratively inserting new control points:

1. Insert new control point(s) into the T-Mesh.
2. If any basis function is missing a knot dictated by Rule 1 for the current T-Mesh, perform the necessary knot insertions into that basis function.
3. If any basis function has a knot that is not dictated by Rule 1 for the current T-Mesh, add an appropriate control point into the T-Mesh.
4. Repeat Steps 2 and 3 until there are no more new operations.

The face with highest L^∞ approximation error is splitted into two rectangles. Knot intervals are updated accordingly (i.e. set to 0.5 for the subdivided edges). We compute the refinements in both directions, and choose the one which performs best in reducing the approximation error. Note that the knot-insertion algorithm may introduce a few additional control points by propagation into the T-Mesh (see Figure 6).

The T-spline local refinement algorithm preserves the geometry of the original T-spline surface. However, our goal is to use these newly introduced degrees of freedom to approximate better the original meshed surface. Therefore, a local fitting process is performed after the local refinement. Note that since the T-Splines function have local degrees of freedom, a smaller linear system needs to be solved. The vector X gathering all the X_i coordinates of the control nodes is split into X_f, the set of control points influenced by the new control point ($f$ree to move) and X_l, the set of control points that will remain $l$ocked. The new degrees of freedom and coupling terms on the boundary of the refined patch are determined by the sparsity pattern of the matrix A. The fitting term is given by:

$$F_{fit}(X_f) = \left\| [A_f | A_l] \begin{bmatrix} X_f \\ X_l \end{bmatrix} - b \right\|^2$$

where A is split into A_f and A_l according to X_f and X_l. The new degrees of freedom are then given by the solution of the linear system:

$$A_f^t A_f X_f = A_f^t A_l X_l - A_f^t b$$

The terms introduced by the fairing energy have the same structure. Since we have a small number of coefficients and since the location of the new control points is not far away from the optimum, we use a conjugate gradient algorithm, that converges in a few iterations.

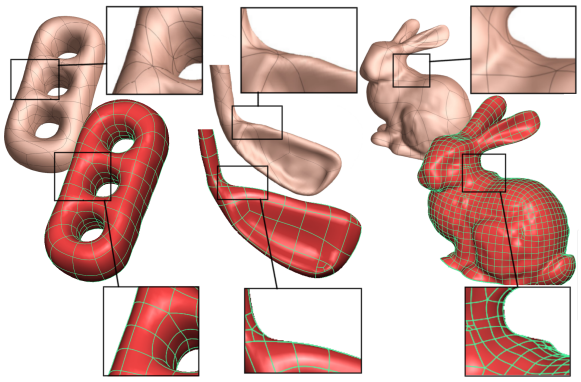


Figure 7: *Comparison between the results of [EH96] and ours: note how the symmetry and anisotropy are respected by our approach.*

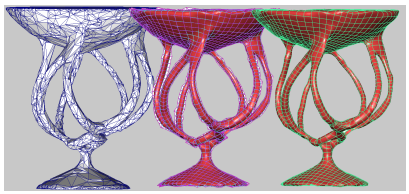


Figure 8: *Our method applied to a high-genus object. From left to right: initial mesh, fitted control mesh and surface. This example also shows the robustness of our method to mesh with poor quality.*

5. Results and Conclusions

Figure 7 compares Eck and Hoppe's results with ours (note that Eck and Hoppe's images reproduced here only show patch boundaries, each patch has a 4x4 control nodes array, therefore control mesh sizes are comparable). As can be seen, our method better respects the symmetries (see e.g. the three-holes torus) and the anisotropy of the objects, as a designer would do. As a consequence, the resulting surfaces do not have wrinkles (see closeups). We show in Figure 8, 9 and 10 our method applied to data sets of various topologies and geometries. For all these examples, less than 15 edges were added by the user. Note that the rocker is topologically equivalent to a torus. Therefore, it would be possible to create a control mesh without any singularity. However,

	No. of vertices	Control nodes	Control nodes (locally refined)
rocker	23k	2021	3692
botijo	41k	1471	2644
horse	10k	2046	2724
vase	2k	2120	2891

Table 1: *Number of control nodes for various models.*

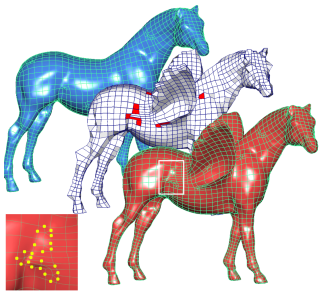


Figure 9: *Conversion from classical mesh models and interactive editing of T-Splines in Maya. The closeup shows how N-sided facets are replaced with T-NURCCes. We also show how this facilitates model editing (pasting the wings of the gargoyle onto the horse).*

we think that the control mesh constructed by our method is a more natural, since it better takes the geometry of the object into account. This is even more obvious in the scanned hand shown in Figure 2. Since the surface is a topological disk, the control mesh could have no extraordinary vertex, but any skilled designer would create one at each finger tip, as our algorithm does, to better balance distortions and adapt the geometry.

Moreover, The *global* energy minimized by PGP overcomes the uneven placement of stream lines obtained with the *local* seeding strategy used in [ASD*03].

Table 1 gives the number of control points obtained for all the models, without and with adaptive local refinement (the L^∞ threshold was set to 0.2% of the bounding box diagonal). As far as timings are concerned, the adaptive fitting algorithm did converge in less than one minute for all these models.

Conclusion: In this paper, we have proposed a method for automatic and interactive mesh to T-spline conversion. Our algorithm proposes an initial solution, that can be manually refined by the user. Using these automatic and manual tools, a complex model can be converted in less than 15 minutes and loaded in industrial software. This is significantly faster than fully manual solutions existing in commercial software. In future work, we think that a better mathematical characterization of the singularities may even further reduce this time, by leading to a fully automatic solution.

Acknowledgments

We thank the European NoE aim at shape and the INRIA ARC GEOREP for funding this research. The first author would also like to thank Huaiping Yang, Weiming Dong, Tom Finnigan, David Cardon and Mattew Sederberg for their help during the development of this work.

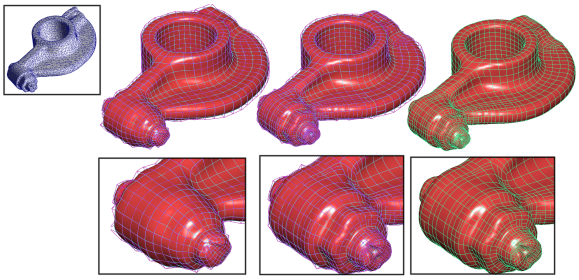


Figure 10: *Converting a scanned mesh into a T-Spline. From left to right: L^2 fitting, L^∞ fitting with local refinement (with and without the control mesh super-imposed).*

References

[aim] AIM AT SHAPE:. http://shapes.aim-at-shape.net/index.php.

[ASD*03] ALLIEZ P., STEINER D. C., DEVILLERS O., LEVY B., DESBRUN M.: Anisotropic Polygonal Remeshing. *ACM TOG (SIGGRAPH)* (2003).

[BBK05] BOTSCH M., BOMMES D., KOBBELT L.: Efficient linear system solvers for mesh processing. In *IMA Mathematics of Surfaces XI, Lecture Notes in Computer Science* (2005).

[BMRJ04] BOIER-MARTIN I., RUSHMEIER H., JIN J.: Parameterization of triangle meshes over quadrilateral domains. In *SGP* (2004), Eurographics.

[Cyb] CYBERWARE:. http://www.cyberware.com/products/cyslice.html.

[d'A00] D'AZEVEDO E.-F.: Are bilinear quadrilaterals better than linear triangles? *SIAM J. of Scientific Computing 22*, 1 (2000), 198–217.

[DBG*06] DONG S., BREMER P.-T., GARLAND M., PASCUCCI V., HART J. C.: Spectral surface quadrangulation. In *SIGGRAPH* (2006).

[DIS03] DODGSON N., IVRISSIMTZIS I., SABIN M.: Curve and surface fitting: Saint malo. In *Nashboro Press, St Malo, France, A. Cohen et al., Eds., vol.2* (2003).

[DKG05] DONG S., KIRCHER S., GARLAND M.: Harmonic functions for quadrilateral remeshing of arbitrary manifolds. In *Computer Aided Geometry Design, Special Issue on Geometry Processing* (2005).

[EH96] ECK M., HOPPE H.: Automatic reconstruction of B-spline surfaces of arbitrary topological type. In *SIGGRAPH* (1996).

[FB88] FORSEY D. R., BARTELS R. H.: Hierarchical b-spline refinement. In *SIGGRAPH conf. proc.* (1988).

[Flo03] FLOATER M. S.: Mean value coordinates. *CAGD 20* (2003), 19–27.

[GGH02] GU X., GOTLER S., HOPPE H.: Geometry images. In *Siggraph* (2002).

[GGT04] GORTLER S., GOTSMAN C., THURSTON D.: *One-Forms on Meshes and Applications to 3D Mesh Parameterization*. Tech. rep., Harvard University, 2004.

[Gre94] GREINER G.: Variational design and fairing of spline surfaces. In *Computer Graphics Forum Volume 13, Issue 3* (1994), pp. 143–154.

[GY03] GU X., YAU S.-T.: Global conformal surface parameterization. In *Symposium on Geometry Processing* (2003), ACM.

[HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *SIGGRAPH* (2000).

[KL96] KRISHNAMURTHY V., LEVOY M.: Fitting smooth surfaces to dense polygon meshes. *Computer Graphics 30*, Annual Conference Series (1996), 313–324.

[KLS03] KHODAKOVSKY A., LITKE N., SCHRODER P.: Globally smooth parameterizations with low distortion. *ACM TOG (SIGGRAPH)* (2003).

[Lev05] LEVY B.: Numerical methods for digital geometry processing. In *Israel Korea Bi-National Conference* (2005).

[LLP05] LI W. C., LEVY B., PAUL J.-C.: Mesh editing with an embedded network of curves. In *Shape Modeling International conference proceedings* (2005).

[LLS01] LITKE N., LEVIN A., SCHRÖDER P.: Fitting subdivision surfaces. In *Proceedings of Scientific Visualization* (2001).

[LMH00] LEE A., MORETON H., HOPPE H.: Displaced subdivision surfaces. In *SIGGRAPH conf. proc.* (2000).

[MBVW95] MILROY M. J., BRADLEY C., VICKERS G. W., WEIR D. J.: G1 continuity of b-spline surface patches in reverse engineering. *Computer-Aided Design 27*, 6 (1995), 471–478.

[MK04a] MARINOV M., KOBBELT L.: Direct anisotropic quad-dominant remeshing. In *Proc. Pacific Graphics* (2004).

[MK04b] MARINOV M., KOBBELT L.: Optimization techniques for approximation with subdivision surfaces. In *Symposium on Solid Modeling and Applications* (2004), ACM.

[Rap] RAPIDFROM:. http://www.rapidform.com/.

[RLL*06] RAY N., LI W. C., LEVY B., SHEFFER A., ALLIEZ P.: Periodic global parameterization, 2006. Accepted pending revisions.

[Sab] SABIN M.:. http://www.cl.cam.ac.uk/~mas33/challenges.htm.

[SCF*04] SEDERBERG T. W., CARDON D. L., FINNIGAN G. T., NORTH N. S., ZHENG J., LYCHE T.: T-spline simplification and local refinement. *ACM TOG (SIGGRAPH)* (2004).

[SdS01] SHEFFER A., DE STURLER E.: Parameterization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers 17* (2001), 326–337.

[SPPH04] SCHREINER J., PRAKASH A., PRAUN E., HOPPE H.: Inter-surface mapping. *ACM TOG (SIGGRAPH)* (2004).

[SZBN03] SEDERBERG T. W., ZHENG J., BAKENOV A., NASRI A. H.: T-splines and T-NURCCs. *ACM TOG (SIGGRAPH)* (2003).

[SZSS98] SEDERBERG T. W., ZHENG J., SEWELL D., SABIN M.: Non-uniform subdivision surfaces. In *SIGGRAPH* (1998).

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: Polycube-maps. *ACM TOG (SIGGRAPH)* (2004).

[ZWS05] ZHENG J., WANG Y., SEAH H. S.: Adaptive t-spline surface fitting to z-map models. *GRAPHITE 2005* (2005).